MS-9A69 (E3845)

Linux GPIO Kernel Module

User Guide

Ver. 0.1

7/31/2018

Prepared by MSI IPC

Document Revision History

Date	Version	Modifier	Remark
2018/7/31	0.1	Mike Chan	Initial version

Content

1. Driver Specification

This driver only support IOCTL method. Direct read and write to the user space device are not supported. Please refer to chapter 5 for how to use the driver.

2. File list

This GPIO kernel module package **GPIO_DRV.tar.xz** includes:
(a). A kernel object file (.ko): **GPIO_DRV.ko**
(b). A header file for use the driver: **GPIO_DRV.h**
(c). A sample C source code for using the driver: **app_sample.c**
(d). Two shell script and a service file for Driver installation: **install.sh**, **uninstall.sh**
(e). A Installation guide: **DriverInstallStep**

3. Install GPIO kernel module

To use the gpio function, a kernel module must installed.

- Change to root account
  ```
  $ su
  ```

- Open a terminal, extract the file by the following command.
  ```
  $ tar xvJf GPIO_DRV.tar.gz
  ```

- Switch the work directory to the kernel module extracted folder.
  ```
  $ cd GPIO_DRV
  ```

- Run ./install.sh with root privilege.
  ```
  $./install.sh
  ```

- A user space device **gpio_drv** will show in /dev with the following permission.
  ```
  $ ls –al /dev/gpio_drv
      crw-rw-rw- 1 root root 247, 0 Jul 30 23:21 /dev/gpio_drv
  ```

Supported Linux Kernel version:

Linux kernel 4.9.0-7-amd64 (Debian 9.5.0 64-bit)

4. Use GPIO function

This chapter is about how to use the gpio function.

- Use the **testapp** with driver package.
  - Get current GPI 3 input level:

```
$ ./testapp /getgpio 3
     get pin_number: 3
     get value: false (input is low)
```

  - Set GPO 2 output level to high:

```
$ ./testapp /setgpio 2 1
     set pin_number: 2
     set value: 1          (set output to high)
```

  - Set GPO 1 output level to low:

```
$ ./testapp /setgpio 1 0
     set pin_number: 1
     set value: 0          (set output to low)
```

5. IOCTL Details

There are 2 IOCTL control code supported in the driver:

IOCTL_GPIOGETVAL: used for get GPI(0~3) value.

IOCTL_GPIOSETVAL: used for set GPO(0~3) value.

The `gpiodrv_arg` struct is as follows.

```
struct gpiodrv_arg {
     unsigned char gpio;  // GPI or GPO value: 0~3
     bool value;          // Hi:true or Low:false
};
```

First, include the header file `"GPIO_DRV.h"`, then open device file `"/dev/gpio_drv"` with `"O_RDWR"` attributes and fill in the `"gpiodrv_arg"` data. Finally, use ioctl to access GPIOs. Please refer to the **"Appendix: app_sample.c"** for detail demonstration.

Appendix: app_sample.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <sys/ioctl.h>
#include "GPIO_DRV.h"

void printCommand(){
     printf("Syntax:\n");
     printf("%s%s",
        "\ttestapp /setgpio pin_num value(1,0)\n",
        "\ttestapp /getgpio pin_num\n");
}

int main(int argc, char *argv[]){
     int pin_number, value;
     bool ret;
     int handle;
     char *error;
     struct gpiodrv_arg gpiodata;

     handle = open("/dev/gpio_drv", O_RDWR);
     if(!handle){
          printf("Error in open gpio driver");
          exit(-1);
     }

     memset(&gpiodata, 0, sizeof(gpiodata));

     if(argc == 1){
          printf("Command Error!\n");
          printCommand();
          close(handle);
          return 0;
     }

     if(strcmp(argv[1],"/setgpio") == 0)
     {
          int level=0;
          ret=false;

          if(argc != 4){
               printf("Syntax Error!\n");
               printCommand();
               close(handle);
               return 0;
```

```
            }

            printf("set pin_number: %s\n", argv[2]);
            printf("set value: %s\n", argv[3]);

            gpiodata.gpio  = atoi(argv[2]);
            gpiodata.value = atoi(argv[3])? true : false;

            ret = ioctl(handle, IOCTL_GPIOSETVAL, &gpiodata);
            if (ret == -EFAULT)
                perror("set value failed\n");

        }
        else if(strcmp(argv[1],"/getgpio") == 0){
            if(argc != 3){
                printf("Syntax Error!\n");
                printCommand();
                close(handle);
                return 0;
            }

            printf("get pin_number: %s\n", argv[2]);
            gpiodata.gpio  = atoi(argv[2]);
            ret = ioctl(handle, IOCTL_GPIOGETVAL, &gpiodata);

            if(ret != -EFAULT)
                printf("get value: %s\n", (gpiodata.value?"true":"false"));
            else
                printf("get value: error\n");

        }
        else
        {
            printf("Unknown Command.\n");
            printCommand();
        }

        close(handle);
        return 0;
}
```